A CS Unplugged Design Pattern

Tomohiro Nishida
Osaka Gakuin University, Japan
nishida@ogu.ac.jp

Susumu Kanemune
Hitotsubashi University, Japan
kanemune@acm.org

Yukio Idosaka
Iinan Junior High School, Japan
idosaka@gmail.com

Mitaro Namiki
Tokyo University of Agriculture and Technology, Japan
namiki@cc.tuat.ac.jp

Tim Bell
University of Canterbury, New Zealand
tim.bell@canterbury.ac.nz

Yasushi Kuno
University of Tsukuba, Japan
kuno@gssm.otsuka.tsukuba.ac.jp

ABSTRACT

"Computer Science (CS) Unplugged" is an educational method for introducing non-specialists to concepts of CS through hands-on activities that don't require the use of a computer. Often the deeper concepts of CS have been considered as being too difficult for elementary and middle school students, and many educators teaching "IT" are not even aware of the richness of the topic. CS Unplugged methods have been used successfully with students of a wide range of ages. In this paper, we analyze the structure of CS Unplugged activities to identify the elements that make them work well. Based on the analysis, we propose a design pattern which will be useful as a guideline for developing new activities, and to revise existing ones. We also describe our experience developing original teaching material, using the pattern as a benchmark for evaluation.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education

General Terms

Design

Keywords

Kinaesthetic learning, pedagogical design patterns

1. INTRODUCTION

In the USA there has been a rapid decline in interest from college students enrolling in Computer Science (CS); this has been reflected in many other countries, or if numbers have not decreased then in some cases the quality of the students has. This decrease in interest can largely be attributed to a gap between the perception of what incoming students think that CS is, and what graduates find [1]. Many approaches have been proposed to address this issue, and one that has met with some success is the CS Unplugged [2, 3] program.

A significant number of the activities have been translated into at least 12 languages [4, 5], and they are being used in even more countries. For example, in Japan a number of teachers and researchers have reported positive responses from students to this approach [6, 7, 8, 9]. Many of the activities have been recommended in the ACM K-12 curriculum [10]. Given this widespread uptake, it will be worthwhile to identify the essence of this approach to encourage the design of new activities, and to identify improvements to existing ones. This is the main purpose of this paper.

In CS Unplugged, students from elementary school ages upwards work *without computers* with hands-on activities that help them to understand a broad range of CS topics in an engaging and motivating way. Unplugged activities have been published in a number of formats, including a book of 20 activities [3], and a more teacher-oriented book of 12 activities. The latter book has been translated and published in Korean [4] and Japanese [5]. The reader is referred to the online resources for more details, but as an example, young students can be introduced to Finite State Automata with a simple but engaging activity that involves running between "islands" (people stationed around the playground) discovering paths in a map (transitions in the FSA).

As CS Unplugged activities have been quite successful so far, we wanted to develop more such activities. However, we wanted to retain the favorable properties of the original activities in new ones. For this purpose, we needed some "design guidelines," which lead to applying the idea of design patterns. *Design patterns* in CS are general reusable solutions to a commonly occurring problem in software design [11]. *Pedagogical* design patterns are the equivalent for the design of a course or lesson. Many patterns have been identified (e.g. Bergin's 14 patterns [12]), but none correspond directly to the approach taken in the CS Unplugged activities. Begel's "Kinesthetic Learning Algorithm" [13] web site provides a template for activities that is called a design pattern, but it doesn't give detailed guidance on how to come up with a new pattern.

In this paper we analyze the fundamental structure and design of the published Unplugged activities with the intention of crystallizing the aspects that make the activities engaging. We then propose a "CS Unplugged Pattern," which provides design principles and guidelines for such educational materials. We hope that new activities can be designed and evaluated using this pattern, so that teachers can develop their own teaching materials. The pattern also highlights weaknesses in some of the published activities, which has led enhancements that we describe and evaluate in Section 3.

2. ANALYSIS OF "CS UNPLUGGED"

In this section we identify the distinctive features of CS Unplugged activities, and propose a pedagogical design pattern that captures these.

2.1 Distinctive features of CS Unplugged

The ACM K-12 curriculum [10] has a strong emphasis on the intellectual and problem-solving nature of CS, which we agree with. The Unplugged approach is particularly aimed at jumping to the heart of CS, avoiding barriers such as having to learn a programming language, being interested in computers, or even having access to them. Unplugged is primarily focused on *outreach* rather than *teaching*, exposing students to ideas to remove misconceptions about the field. The main characteristics of the Unplugged activities are:

No computers: By definition, computers are not used directly in the activities. This immediately removes programming ability or even owning a computer as a prerequisite for doing CS. Of course, inevitably computers are used to develop and publish the materials.

Games: The activities are generally based around a game or challenge, so that children see them as play, which leads to interest, curiosity and motivation.

For example, in the "Card Flip Magic" activity, the surprise of the magic trick attracts the interest of the students, and their desire to find out the trick provides motivation to understand what is happening. In the "Battleship" activity, a game with an opponent attracts the students' interest, and their desire to win the game provides their motivation to seek a strategy (or algorithm) appropriate for that game. In the "Treasure Hunt" activity, the motif of a treasure island attracts the students' interest, and letting the students draw their own trails naturally leads to understanding the idea of finite state automata.

Kinaesthetic: Physical objects are used, such as cards and weights. This provides kinaesthetic engagement, and often introduces humor if the objects are out of context (such as fruit being used in a computing class).

Student directed: The activities generally involve interaction with other students, and encourage students to discover answers by trial and error. This engages them with the problems, and demonstrates that they can discover great ideas for themselves. Working with other students encourages teamwork and communication, also important traits of a CS graduate.

Easy implementation: The activities are easy to prepare and use only inexpensive equipment, much of which can be found in a school. Where handouts are needed, material is provided so that teacher can simply copy it and not have to prepare resources.

Growing body of ideas: Sharing ideas and variations of activities is an important aspect of the project, and many clever variations of activities have developed as educators around the world have used them and adapted them to local conditions.

Sense of story: Often elements of fantasy and story telling are used to engage students (e.g. pirates, secret messages). This can draw younger students into the activity, and emphasises the value of creativity rather than dry learning.

It is customary to use computer rooms for computing and informatics classes. Because CS Unplugged activities do not use computers, so they can be carried out in ordinary classrooms or even in outdoors. In fact, trying to run them in a computer room can be difficult, as students expect to be using a computer for CS, and can see the activities as just a pre-amble to getting onto the computers, reducing their concentration on the activities.

2.2 A "CS Unplugged Pattern"

Design patterns are collections of good design rules and essences, and originate from the architectural patterns (for towns and buildings) by C. Alexander [14] in the late 1970's. In the domain of CS, the "Gang of Four patterns [11]" and "analysis patterns [15]" are well-known.

Since this early work there has been much work around patterns, with collections being created for various aspects of software development, and various books and conferences such as PLoP (Pattern Languages of Programming) [16] are available. We have chosen J. Coplien's pattern template [17] as the basis of our pattern description. The goal is to make it easy to develop appropriate teaching materials. Our CS Unplugged Pattern is shown in Figure 1, and a commentary is provided below.

Pattern Name

Assign an appropriate name. We want to preserve the reference to the model of the CS Unplugged materials, so the pattern have been named "CS Unplugged Pattern."

Problem

Explanation of the problem that must be solved. The main issue is the lack of understanding in many schools of what CS is.

Context

The context or situation in which the solution is applicable. An important age for children making career decisions is around 12 years old, when they are making decisions about what specializations they will develop through their high school years after leaving a generalist elementary school education. CS Unplugged covers a wide range of ages, but is particularly targeted for the children in their late elementary years (ages 9–12), and should also be applicable to junior high school students (ages 12–15) at least.

Forces

The conditions in which this pattern is applicable. A fundamental principle is that the material can be taught without computers. However, we also want to ensure that the activities are useful for school classes, which is

Table 1: Example mappings from everyday objects to CS concepts

Objects	Mappings
Cards	Two states (two sides), choice (drawing from a shuffled deck), combinations and permutations (shuffling, spreading out)
Cups, containers, buckets	Hidden information (on top of another object), two states (normal, inverted), variable (contains another object), limited contents
Stickers, marker pen	Commitment (can't backtrack if permanent), backtracking (if not permanent), label (on another object), range of values (colour, characters), user input (writing text or a number, marking a choice)
Chalk on pavement, tape on floor	Transitions, edges, links, nodes, vertices, paths, target (at the end of paths)
Board game	Paths (following layout on the board), chance (rolling dice, choosing cards), rules (limited number of operations)
Food	Competition (to win a candy/chocolate), humor (using food out of context), color (of fruit, candies), sharing (dividing a cake or chocolate), size (comparing items)
String	Connection, communication (pull on the string, slide a message along), edge weights (length of string), network

why the forces include making the material accessible to non-specialist teachers, and able to be completed within the framework of a school day. This length prescribed also relates to the concentration span of school children, and also enables the activities to be used in other situations, such as an out-of-school club.

Solution

The strategy to solve the problem. This is the key component for constructing new activities. Here we provide a series of steps for constructing an exercise. Of course, like any creative activity, there is no simple formula, iteration over these steps will be required, and some activities can be designed simply based on existing puzzles and games that happen to map to CS concepts (such as the well-known "knitting needle" sorting and searching [18]).

1. Choose a concept from CS to be communicated.
2. Identify the key elements of the concept, such as bits (in data representation), states and transitions (in finite state machines), edges and vertices (in graphs), comparison (in sorting and searching), pixels (in graphics), psychological phenomena (in HCI), or relationships (in graphs and other data structures).
3. Consider what games, puzzles, toys, or common objects use similar elements. Table 1 gives some example mappings of CS concepts to physical items. A visit to places like stationery stores, bargain basements, playgrounds, educational suppliers and toy stores can be useful for getting ideas.
4. *Turn it into a challenge.* An important goal is for students to discover ideas for themselves, not to teach them a particular algorithm or technique for its own sake. Thus the activity needs to contain a challenge that will engage the students, such as trying to find a lower cost solution, or complete the challenge before another group. Consider using a team activity where students can engage in different roles, and a successful outcome means success for the team. This will prompt spontaneous communication among students and encourages them to exchange ideas, which leads to deeper thinking and more insight. Making a challenge can also involve deliberately including an impediment, such as one participant not being able to see all the information (e.g. in the battleship game), or restricting the way in which objects can be manipulated (e.g. in the sorting algorithms where only two weights can be compared at a time) — "I hear and I forget, I see and I remember, I do and I understand."
5. *Evaluate.* There is no substitute for testing activities with a variety of students. Sometimes this exposes flaws, and sometimes the students come up with improvements or simplifications that improve the activity. The evaluation must consider the simplicity, engagement level, cost and novelty of the activity. It is also useful to have other teachers use it, and have other experts in that domain look at it to see if there is some element of the domain that could be added to improve the activity.
6. *Refine.* This follows from the evaluation.
7. *Publish.* The material should be published as a ready-to-use handout including information on how to obtain resources, and the applications of the concept being demonstrated. Preparing resources to explain the actual use (of the principle) in computers is important because students can link their experiences to technologies in the real world. Because teachers may be non-specialists, they many not have the background to give examples themselves from real-world applications, so accompanying resources should enable them to understand this *and* present it to the students.

Resulting Context.

The obvious goal is to increase the breadth and quality of activities available so that educators can choose one that is appropriate to the topic at hand, or helps them to illustrate a concept that they wish to demonstrate so that students better understand what CS is.

Rationale.

An explanation of the value of the CS Unplugged Pattern has already been given in Section 1.

The CS Unplugged pattern is intended to be an easy and effective tool for developing original teaching materials that fulfill the concepts and merits of CS Unplugged. Moreover, existing CS Unplugged materials can be reviewed and verified using the pattern so that weak points can be discovered and their quality can be enhanced. Additionally, materials developed independently from CS Unplugged can be checked against the pattern to discriminate those that have desirable characteristics.

In the following section, we introduce some original activities developed as an extension of those in the CS Unplugged textbooks, and we evaluate the material using CS Unplugged Pattern.

Pattern Name: CS Unplugged Pattern
Problem:

- It is difficult to communicate deep CS principles to students who have no background in programming or access to computers.

Context:

- Students without specific CS knowledge (main from ages from 7 to 15 and up) should be able to understand and appreciate the material

Forces:

- Needs to be taught without computers.
- Teachers who are not professional CS researchers can teach the material.
- The material can be taught in one lesson (typically 20 to 40 minutes).

Solution:

- Choose a concept from CS to be communicated.
- Identify the key elements.
- Consider what games, puzzles, toys or common objects use similar elements.
- Turn it into a challenge.
- Evaluate the activity with students.
- Refine the activity based on the evaluation.
- Publish the activity, with information on obtaining resources, and with an explanation of the relevance of the concepts being demonstrated.

Resulting Context:

- Students understand the concept from CS.
- More concepts from CS are covered.

Rationale:

- Group-work and games enhance students' motivation and engage them to think about CS principles.
- Teaching resources are specifically targeted to teach the essence of CS principles even if the teacher is not familiar with the concept.

Figure 1: CS Unplugged Pattern

3. DEVELOPMENT AND EVALUATION OF NEW MATERIAL

There is a need to expand the coverage of the Unplugged activities so that a wider range of topics is available, and so that existing topics can be explained in more detail.

For example, in the "Card Flip Magic" activity, students can understand that a parity check can detect and correct errors, but the need for this may not be immediately apparent. This is problematic for younger students, for whom the activity is the main experience, and explanations may not have such a great impact. It is also useful to provide a richer experience for high school students to help them understand real-world applications of the CS principles they have learned in the activities.

Therefore, we have developed the "Telephone Game" to enhance students' understanding of why error detection and correction is required, and have evaluated it at Osaka Gakuin high school. This activity was developed concurrently with CS Unplugged Pattern, so many of its characteristics naturally follow the pattern.

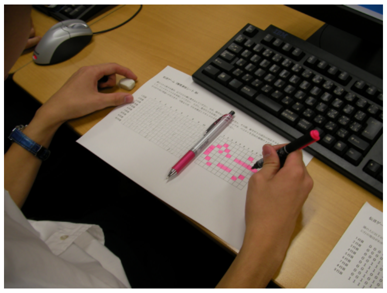


Figure 2: A student playing the "Telephone Game"

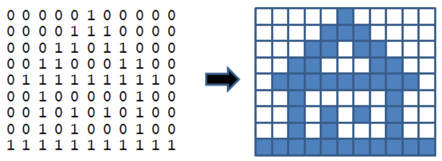
The "Telephone Game" is carried out as follows:

1. Students are organized in groups, and the first student in the group receives the worksheet (Figure 3). In the worksheet, a picture of an 11 by 9 bitmap is encoded in a 0/1 sequence.
2. The student should transfer the 0/1 sequence to the next student. They are instructed to speak each digit only once, with no correction.
3. Each student receives the 0/1 sequence and transfers it to the next student. However, when they are using parity check (as in the "Card Flip Magic,") they are allowed to correct the faulty bit.
4. The last student in each group reconstructs the bitmap picture.

An evaluation was performed by participating in a joint university-high school project, teaching programming classes to third (the highest) grade students (aged around 17-18), from April 2007 to December 2007. 10 students attended the class, which worked through four Unplugged activities, plus the "telephone game".

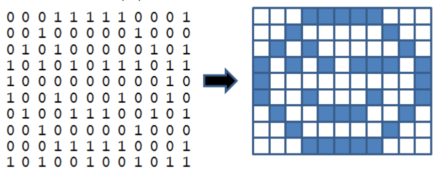
We organized the 10 students into 3 teams (Team A: 3, Team B: 3, Team C: 4), and conducted the game first without parity and then with parity. The direction of transfer was reversed between the games.

As shown in Table 2, with parity two groups could transfer the image without error, and the remaining team also had only 1 error. Even if we consider the learning effect, the usefulness of parity was clear to all present. In a survey of the class, 6 students reported understanding the role and usefulness of a parity check, 3 understood "somewhat", and one reported only partial understanding.



```
0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 0 1 1 0 1 1 0 0 0
0 0 1 1 0 0 0 1 1 0 0
0 1 1 1 1 1 1 1 1 1 0
0 0 1 0 0 0 0 0 1 0 0
0 0 1 0 1 0 1 0 1 0 0
0 0 1 0 1 0 0 0 1 0 0
1 1 1 1 1 1 1 1 1 1 1
```

(a) without parity



```
0 0 0 1 1 1 1 1 0 0 0 1
0 0 1 0 0 0 0 0 1 0 0 0
0 1 0 1 0 0 0 0 0 1 0 1
1 0 1 0 1 0 1 1 1 0 1 1
1 0 0 0 0 0 0 0 0 0 1 0
1 0 0 1 0 0 0 1 0 0 1 0
0 1 0 0 1 1 1 0 0 1 0 1
0 0 1 0 0 0 0 0 1 0 0 0
0 0 0 1 1 1 1 1 0 0 0 1
1 0 1 0 0 1 0 0 1 0 1 1
```

(b) with parity

Figure 3: The data used in "Telephone Game"

Table 2: the result of "Telephone Game"

Team	1st (without parity)	2nd (with parity)
A	Fail(19 errors)	Fail (1 error)
B	Fail(6 errors)	Success
C	Fail(2 errors)	Success

Another extension activity, "Creating and recovering maps" was developed to extend the "Treasure Island" (Finite state automata) activity. In this, they created their own FSA and wrote programs to accept those. After that they checked each other's programs and recovered maps in pairs.

At the end of the semester, the students were asked to identify the activities the students that they thought were interesting. The "Telephone Game" was chosen by the most (7 students out of 9), with "Battleships" and "Creating and Recovering Maps" being second (6 students each). This indicates that the activities that build on others and explain their application are the more engaging for high school students.

We have checked those activities against CS Unplugged Pattern, and verified that the activities conforms to most of the properties (actually the teacher was a professional CS researcher, but the high school teacher attended to the class confirmed that he will be able to use the activities satisfactorily).

In contrast, a programming activity ("Look-up Program") based on the battleships searching game was not mentioned by the students. When verified against CS Unplugged Pattern, the activity does not conform to the pattern in that: (1) it uses computers, (2) it requires some CS background, (3) it is not a game or a challenge.

This result suggests the usefulness of the CS Unplugged Pattern, but also raises the question of how Unplugged activities might better be linked to online activities.

4. CONCLUSION

In this paper, we have analysed "CS Unplugged" materials in which students can learn principles of CS without computers. From the analysis, we have proposed a "CS Unplugged Pattern" as a guideline for developing teaching resources, and a methodology for identifying components that might be used to construct a kinesthetic activity.

Although CS Unplugged materials relate to abstract concepts of CS, the use of demonstrations and game experiences allow the students to learn and understand those concepts in an entertaining way and with high motivation. With the CS Unplugged Pattern, we can expect to develop more teaching resources with these desirable characteristics.

5. REFERENCES

[1] Rick Rashid: Inspiring a New Generation of Computer Scientists, Communications of the ACM, Vol.51, No.7, pp.33–34, 2008.

[2] Tim Bell, Ian H. Witten, Mike Fellows: Computer Science Unplugged – An enrichment and extension programme for primary-aged children, 2005. `http://csunplugged.com/`

[3] Tim Bell, Ian H. Witten, Mike Fellows: Computer Science Unplugged: off-line activities and games for all ages, 1999.

[4] Lee WonGyu (translation): Computer Science Unplugged (Korean Version), Hongreung Science Publishing, 2006.

[5] Susumu Kanemune et al. (translation): Computer Science Unplugged (Japanese Version), Etext, 2007.

[6] Tomohiro Nishida et al.: New Methodology of Information Education with "Computer Science Unplugged", ISSEP 2008 Proceedings, LNCS 5090, Springer, pp.241–252, 2008.

[7] Yukio Idosaka et al.: A Practical Approach for Elementary Schoolchildren with "Computer Science Unplugged" , Proceedings of SSS2008, pp.25–32, 2008. (In Japanese)

[8] Yayoi Hofuku et al.: Using CS Unplugged in High School Information-B Classes, Proceedings of SSS2008, pp.201–206, 2008. (In Japanese)

[9] Hiroki Manabe et al.: Information Education in a Vocational Training School for Persons with Disabilities, Proceedings of SSS2008, pp.171–178, 2008. (In Japanese)

[10] ACM K-12 Task Force Curriculum Committee: ACM K-12 CS Model Curriculum, 2nd Edition, 2003. `http://csta.acm.org/Curriculum/sub/ACMK12CSModel.html`

[11] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.

[12] Joseph Bergin: Fourteen Pedagogical Patterns. `http://csis.pace.edu/~bergin/PedPat1.3.html`

[13] Andrew Begel, Daniel D. Garcia and Steven A. Wolfman: Kinesthetic learning in the classroom, SIGCSE Bull., 36(1), 183–184, 2004.

[14] Christopher Alexander, Sara Ishikawa, and Murray Silverstein: A Pattern Language: Towns, Buildings, Construction, Oxford University Press, 1977.

[15] Martin Fowler: Analysis Patterns: Reusable Object Models, Addison-Wesley, 1996.

[16] Pattern Languages of Programs. `http://hillside.net/conferences/plop.htm`

[17] James Coplien: Software Patterns, 1996. `http://sites.google.com/a/gertrudandcope.com/info/Publications/Patterns/WhitePaper`

[18] Paul Curzon: Computing Without Computers. `http://www.dcs.qmul.ac.uk/~pc/research/education/puzzles/reading/`